Scaling Up Feature Selection: A Distributed Filter Approach

Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Joana Cerviño-Rabuñal

Laboratory for Research and Development in Artificial Intelligence (LIDIA), Computer Science Dept., University of A Coruña, 15071 A Coruña, Spain
{veronica.bolon,noelia.sanchez,joana.cervino}@udc.es

Abstract. Traditionally, feature selection has been required as a preliminary step for many pattern recognition problems. In recent years, distributed learning has been the focus of much attention, due to the proliferation of big databases, in some cases distributed across different nodes. However, most of the existing feature selection algorithms were designed for working in a centralized manner, i.e. using the whole dataset at once. In this research, a new approach for using filter methods in a distributed manner is presented. The approach splits the data horizontally, i.e., by samples. A filter is applied at each partition performing several rounds to obtain a stable set of features. Later, a merging procedure is performed in order to combine the results into a single subset of relevant features. Five of the most well-known filters were used to test the approach. The experimental results on six representative datasets show that the execution time is shortened whereas the performance is maintained or even improved compared to the standard algorithms applied to the non-partitioned datasets.

1 Introduction

In the past 20 years, the dimensionality of the datasets involved in data mining has increased dramatically, as can be seen in [1]. This fact is reflected if one analyzes the *dimensionality* (samples × features) of the datasets posted in the UC Irvine Machine Learning Repository [2]. In the 1980s, the maximal dimensionality of the data was about 100; then in the 1990s, this number increased to more than 1500; and finally in the 2000s, it further increased to about 3 million. The proliferation of this type of datasets with very high ($>$ 10000) dimensionality had brought unprecedented challenges to machine learning researchers. Learning algorithms can degenerate their performance due to overfitting, learned models decrease their interpretability as they are more complex, and finally speed and efficiency of the algorithms decline in accordance with size.

Machine learning can take advantage of feature selection methods to be able to reduce the dimensionality of a given problem. *Feature selection* (FS) is the process of detecting the relevant features and discarding the irrelevant and redundant ones, with the goal of obtaining a small subset of features that describes properly the given problem with a minimum degradation or even improvement

C. Bielza et al. (Eds.): CAEPIA 2013, LNAI 8109, pp. 121–130, 2013.

in performance [3]. Feature selection, as it is an important activity in data preprocessing, has been an active research area in the last decade, finding success in many different real world applications [4,5,6,7].

FS methods usually come in three flavors: *filter*, *wrapper*, and *embedded* methods [8]. The *filter* model relies on the general characteristics of training data and carries out the FS process as a pre-processing step with independence of the induction algorithm. On the contrary, *wrappers* involve optimizing a predictor as a part of the selection process. Halfway these two models one can find *embedded* methods, which perform FS in the process of training and are usually specific to given learning machines. By having some interaction with the predictor, wrapper and embedded methods tend to obtain higher prediction accuracy than filters, at the cost of a higher computational cost. When dealing with high dimensional data, as in this research, filters are preferable even when the subset of features is not optimal, due to their computational and statistical scalability [9].

Traditionally, FS methods are applied in a centralized manner, i.e. a single learning model to solve a given problem. However, when dealing with large amounts of data, distributed FS seems to be a promising line of research since allocating the learning process among several workstations is a natural way of scaling up learning algorithms. Moreover, it allows to deal with datasets that are naturally distributed, a frequent situation in many real applications (e.g. weather databases, financial data or medical records). There are two common types of data distribution: (a) horizontal distribution wherein data are distributed in subsets of instances; and (b) vertical distribution wherein data are distributed in subsets of attributes. The great majority of approaches distribute the data horizontally, since it constitutes the most suitable and natural approach for most applications [10,11,12,13]. While not common, there are some other developments that distribute the data vertically [14,15,16]. When the data come distributed in origin, vertical distribution is solely useful where the representation of data could vary along time by adding new attributes.

In this research, and in order to deal with large databases, we will distribute the data horizontally. In this manner, several rounds of FS processes will be performed, whose outputs will be combined into a single subset of relevant features. Experimental results on six benchmark datasets demonstrate that our proposal can maintain the performance of original FS methods, providing a learning scalable solution.

The rest of the paper is organized as follows: Section 2 presents our distributed filter approach, Section 3 depicts the experimental setup, and Sections 4 and 5 report the experimental results and the conclusions, respectively.

2 Distributed Feature Selection

In this paper we present a distributed filter approach by partitioning the data horizontally. The methodology consists of applying filters over several partitions of the data, combined in the final step into a single subset of features. The idea of distributing the data horizontally builds on the assumption that combining

the output of multiple experts is better than the output of any single expert. There are three main stages: (i) partition of the datasets; (ii) application of the filter to the subsets; and (iii) combination of the results.

The feature selection algorithm (see pseudo-code in Algorithm 1) is applied to all the datasets in several iterations or rounds. This repetition ensures capturing enough information for the combination stage. At each round, the first step is the partition of the dataset, which consists of randomly dividing the original training dataset into several disjoint subsets of approximately the same size that cover the full dataset (see Algorithm 1, line 3). As mentioned above, the partition will be doing horizontally . Then, the filter algorithm chosen is applied to each subset separately and the features selected to be removed receive a vote (Algorithm 1, lines 5 - 8). At that point, a new partition is performed and another round of votes is accomplished until reaching the predefined number of rounds. Finally, the features that have received a number of votes above a certain threshold are removed. Therefore, a unique set of features is obtained to train a classifier C and to test its performance over a new set of samples (test dataset).

To determine the threshold of votes required to remove a feature is not an easy-to-solve question, since it depends on the given dataset. Therefore, we have developed our own automatic method which calculates this threshold, outlined in Algorithm 1, lines 9-19. The best value for the number of votes is estimated from its effect on the training set, but due to the large size of the dataset, not the complete training set was used, only 10% was employed.

Following the recommendations exposed in [17], the selection of the number of votes must take into account two different criteria: the training error and the percentage of features retained. Both values must be minimized to the extent possible, by minimizing the fitness criterion $e[v]$ (see Algorithm 1, line 18). To calculate this criterion, a term α is introduced to measure the relative relevance of both values and was set to $\alpha = 0.75$ as suggested in [17], giving more influence to the classification error. Because of performing a horizontally partition of the data, the maximum number of votes is the number of rounds r times the number of subsets s. Since in some cases this number is in the order of thousands, instead of evaluating all the possible values for the number of votes we have opted for delimiting into an interval $[minVote, maxVote]$ computed used the mean and standard deviation (see lines 9-12 in Algorithm 1).

3 Experimental Setup

This section presents the datasets chosen for testing the distributed approach and the concrete filters which will carry out the feature selection process. For testing the adequacy of our proposal, four well-known supervised classifiers, of different conceptual origin, were selected: C4.5, naive Bayes, IB1 and SVM. All the classifiers and filters are executed using the Weka tool [18], with default values for their parameters. Notice that the C4.5 classifier, widely-used in the FS literature, performs its own embedded selection of features so it might be using a smaller number of features than the other ones.

Algorithm 1: Pseudo-code for distributed filter

Data: $\mathbf{d}_{(m \times n+1)} \leftarrow$ labeled training dataset with m samples and n input features

X $\leftarrow$set of features, X $= \{x_1, \ldots, x_n\}$
$s \leftarrow$ number of submatrices of $\mathbf{d}$ with p samples
$r \leftarrow$ number of rounds
$\alpha \leftarrow 0.75$

Result: S $\leftarrow$ subset of features \S $\subset$ X

`/* Obtaining a vector of votes for discarding features */`

initialize the vector votes to 0, |vector|=n
for *each round* **do**
  Split $\mathbf{d}$ randomly into s disjoint submatrices
  for *each submatrix* **do**
    apply a feature selection algorithm
    F $\leftarrow$ features selected by the algorithm
    E $\leftarrow$ features eliminated by the algorithm \E $\cup$ F = X
    increment one vote for each feature in E
  end
end

`/* Obtain threshold of votes,` Th`, to remove a feature */`

$avg \leftarrow$ compute the average of the vector votes
$std \leftarrow$ compute the standard deviation of the vector votes
$minVote \leftarrow$ minimum threshold considered (computed as $avg - 1/2std$)
$maxVote \leftarrow$ maximum threshold considered (computed as $avg + 1/2std$)
$\mathbf{z} \leftarrow$ submatrix of $\mathbf{d}$ with only 10% of samples
for $v \leftarrow$ *mixVote to maxVote with increment 5* **do**
  $\mathrm{F}_{th} \leftarrow$ subset of selected features (number of votes $< v$)
  $error \leftarrow$ classification error after training $\mathbf{z}$ using only features in F_{th}
  $featPercentage \leftarrow$ percentage of features retained $\left(\frac{|\mathrm{F}_{th}|}{|\mathrm{X}|} \times 100\right)$
  $e[v] \leftarrow \alpha \times error + (1 - \alpha) \times featPercentage$
end
$Th \leftarrow min(e)$, Th is the value which minimizes the error e
S $\leftarrow$ subset of features after removing from X all features with a number of votes $\geq Th$

3.1 Datasets

In order to test our distributed filter approach, we have selected six benchmark datasets which are reported in Table 1, depicting their properties (number of features, number of training and test instances and number of classes). These datasets can be considered representative of problems from medium to large size, since the horizontally distribution is not suitable for small-sample datasets. All of them can be free downloaded from the UCI Machine Learning Repository [2]. Those datasets originally divided into training and test sets were maintained,

whereas, for the sake of comparison, datasets with only training set were randomly divided using the common rule 2/3 for training and 1/3 for testing. The number of packets (s) to partition the dataset in each round is also displayed in the last column of Table 1. This number was calculated with the constraint of having, at least, three packets per dataset.

Table 1. Dataset description

Dataset	Features	Training	Test	Classes	Packets
Connect4	42	45038	22519	3	45
Isolet	617	6238	1236	26	5
Madelon	500	1600	800	2	3
Ozone	72	1691	845	2	11
Spambase	57	3067	1534	2	5
Mnist	717	40000	20000	2	5

3.2 Filter Methods

The distributed approach proposed herein can be used with any filter method. In this work, five well-known filters, based on different metrics, were chosen. While three of the filters return a feature subset (CFS, Consistency-based and INTERACT), the other two (ReliefF and Information Gain) are ranker methods, so it is necessary to establish a threshold in order to obtain a subset of features. In this research we have opted for retaining the c top features, being c the number of features selected by CFS. It is also worth noting that although most of the filters work only over nominal features, the discretization step is done by default by Weka, working as a black box for the user.

- **Correlation-based Feature Selection** (CFS) is a simple filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function [19]. Theoretically, irrelevant features should be ignored and redundant features should be screened out.
- The **Consistency-based Filter** [20] evaluates the worth of a subset of features by the level of consistency in the class values when the training instances are projected onto the subset of attributes.
- The **INTERACT** algorithm [21] is based on symmetrical uncertainty (SU). The authors stated that this method can handle feature interaction, and efficiently selects relevant features. The first part of the algorithm requires a threshold, but since the second part searches for the best subset of features, it is considered a subset filter.
- **Information Gain** [22] is one of the most common attribute evaluation methods. This filter provides an ordered ranking of all the features and then a threshold is required.
- **ReliefF** [23] is an extension of the original Relief algorithm that adds the ability of dealing with multiclass problems and is also more robust and capable of dealing with incomplete and noisy data. This method may be applied

in all situations, has low bias, includes interaction among features and may capture local dependencies which other methods miss.

4 Experimental Results

In this section we present and discuss the experimental results over six benchmark datasets. Our distributed approach is compared with the centralized standard approach of each method. To distinguish between both approaches, a "C" (centralized) or a "D" (distributed) was added to the name of the filter. In the case of the distributed approach, three rounds (r in Algorithm 1) have been executed.

Table 2 reports the test classification accuracies of C4.5, naive Bayes, IB1 and SVM over the six datasets. The best result for each dataset and classifier is highlighted in bold face, while the best result for dataset is also shadowed.

As expected, the results are very variable depending on the dataset and the classifier. However, in terms of average (last column), the best result for each classifier is obtained by a distributed approach, except for SVM. In particular, ReliefF-D combined with C4.5 achieves the highest accuracy, outperforming in at least 4% the best results for the remaining classifiers.

For datasets *Connect4* and *Isolet*, the highest accuracies are obtained by centralized approaches, although these results improve only in 0.90% and 2.19%, respectively, the best mark achieved by a distributed method. For Ozone dataset, both distributed and centralized approaches obtain the highest precision when combined with SVM classifier.

For the remaining datasets (*Madelon*, *Spambase* and *Mnist*), the best results are accomplished by a distributed method. It is worth mentioning the case of *Spambase*, where ReliefF distributed combined with naive Bayes reports 91.79% of classification accuracy whilest the same filter method in the standard centralized approach achieves a poor 41.85% of accuracy. The results for *Mnist* dataset are also remarkable, where the highest accuracy (96.31%) outperforms the best mark of a centralized method in more than 6%.

Table 3 reports the runtime of the feature selection algorithms, both in centralized and distributed manners. In the distributed approach, considering that all the subsets can be processed at the same time, the time displayed in the table is the average of the times required by the filter in each subset generated in the partitioning stage. In these experiments, all the subsets were processed in the same machine, but the proposed algorithm can be executed in multiple processors. Please note that this filtering time is independent of the classifier chosen.

As expected, the advantage of the distributed approach in terms of execution time over the standard method is significant. The time is reduced for all datasets and filters. It is worth mentioning the important reductions as the dimensionality of the dataset grows. For *Mnist* dataset, which has 717 features and 40000 training samples, the reduction is more than notable. For ReliefF filter, the processing time is reduced from almost 8 hours to 15 minutes, proving the adequacy of the distributed approach when dealing with large datasets.

Table 2. Test classification accuracy. Best results are highlighted.

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist	Average
C4.5	CFS-C	61.22	81.59	80.50	97.63	81.16	86.99	81.51
	CFS-D	61.25	**82.23**	76.88	95.86	79.27	88.65	80.69
	INT-C	60.48	78.96	80.63	96.92	78.16	87.24	80.40
	INT-D	61.66	79.03	82.38	94.79	80.83	88.62	81.22
	Cons-C	60.49	56.00	80.63	**98.70**	84.62	87.00	77.90
	Cons-D	61.66	77.10	82.63	96.33	79.34	**90.46**	81.25
	IG-C	**63.90**	81.40	72.75	98.22	83.83	87.83	81.32
	IG-D	62.34	81.08	79.63	97.87	**85.33**	87.88	82.36
	ReliefF-C	63.49	79.54	73.88	98.11	78.81	87.34	80.19
	ReliefF-D	63.00	80.56	**87.50**	98.46	84.75	87.95	**83.70**
NB	CFS-C	60.28	**75.05**	**71.75**	78.22	57.69	71.88	69.15
	CFS-D	58.83	73.89	70.13	76.69	57.24	73.34	68.35
	INT-C	53.85	71.26	70.00	78.22	57.95	70.94	67.04
	INT-D	59.16	70.75	70.13	75.03	74.77	71.06	70.15
	Cons-C	54.12	42.78	70.00	**98.70**	91.00	72.78	71.56
	Cons-D	59.16	69.92	70.38	73.25	**92.89**	**75.74**	**73.56**
	IG-C	60.42	69.34	70.38	74.08	76.53	70.74	70.25
	IG-D	60.28	67.54	70.63	77.63	89.70	68.09	72.31
	ReliefF-C	60.42	62.67	68.63	71.36	41.85	69.82	62.46
	ReliefF-D	**60.50**	56.51	71.50	60.95	91.79	70.93	68.70
IB1	CFS-C	53.90	56.00	85.63	96.45	79.14	87.93	76.51
	CFS-D	57.61	54.78	65.63	96.57	77.31	91.65	73.93
	INT-C	58.27	52.92	88.75	94.44	79.73	86.87	76.83
	INT-D	57.61	49.84	71.75	95.27	76.86	91.79	73.85
	Cons-C	58.06	49.90	88.75	**98.70**	**80.83**	87.36	77.27
	Cons-D	57.61	58.31	71.63	95.27	77.38	**96.31**	76.09
	IG-C	51.29	54.78	74.25	95.98	78.62	89.63	74.09
	IG-D	57.01	**59.72**	86.13	95.50	78.42	90.77	77.92
	ReliefF-C	**61.81**	59.14	75.25	95.98	76.99	89.97	76.52
	ReliefF-D	57.01	57.09	**90.88**	96.80	80.70	91.35	**78.97**
SVM	CFS-C	**60.42**	83.45	66.50	**98.70**	**85.85**	79.58	**79.08**
	CFS-D	**60.42**	82.42	67.13	**98.70**	82.27	**81.52**	78.74
	INT-C	**60.42**	73.83	66.38	**98.70**	80.31	78.54	76.36
	INT-D	**60.42**	78.00	**68.50**	**98.70**	81.49	80.84	77.99
	Cons-C	**60.42**	31.17	66.38	**98.70**	81.88	75.14	68.95
	Cons-D	**60.42**	68.12	66.50	**98.70**	81.94	80.85	76.09
	IG-C	**60.42**	82.94	67.13	**98.70**	83.83	78.28	78.55
	IG-D	**60.42**	79.67	67.13	**98.70**	83.38	79.30	78.10
	ReliefF-C	**60.42**	**84.61**	67.50	**98.70**	81.94	75.43	78.10
	ReliefF-D	**60.42**	82.36	67.50	**98.70**	83.57	75.72	78.04

For the distributed approach, there exist also the time required to find the threshold to build the final subset of features. This time highly depends on the classifier, as can be seen in Table 4. In this table it is visualized the average runtime for each filter and classifier. It is easy to note that the classifier which

Table 3. Runtime (hh:mm:ss) for the FS methods tested. Lowest times highlighted in bold font.

Method	Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
CFS-C	00:02:25	00:05:49	00:00:55	00:00:12	00:00:16	00:44:55
CFS-D	00:00:06	00:01:12	**00:00:13**	00:00:04	**00:00:04**	00:05:24
INT-C	00:02:57	00:04:55	00:00:56	00:00:12	00:00:16	00:42:13
INT-D	**00:00:05**	00:00:54	00:00:14	00:00:04	**00:00:04**	00:04:50
Cons-C	00:13:36	00:07:03	00:01:01	00:00:12	00:00:19	03:22:21
Cons-D	**00:00:05**	00:01:02	00:00:14	00:00:04	00:00:05	00:09:37
IG-C	00:02:19	00:04:32	00:00:55	00:00:12	00:00:16	00:38:17
IG-D	**00:00:05**	**00:00:49**	**00:00:13**	**00:00:03**	**00:00:04**	**00:04:46**
ReliefF-C	00:31:40	00:13:04	00:01:23	00:00:14	00:00:29	07:54:40
ReliefF-D	00:00:06	00:00:57	00:00:17	**00:00:03**	**00:00:04**	00:15:59

requires more execution time is SVM whilst the one which requires the shortest time is naive Bayes. In any case, this is usually in the order of seconds (2 minutes in the worst case) so it is insignificant when compared with the time required by any of the centralized algorithms showed above. Moreover, if the user would rather save this time, it is possible to establish a fixed threshold and not performing this specific calculation.

Table 4. Average runtime (hh:mm:ss) for obtaining the threshold of votes. Lowest times highlighted in bold font.

Method	C4.5	NB	IB1	SVM
CFS-D	00:00:36	00:00:26	00:00:48	00:01:36
INT-D	00:00:31	00:00:24	00:00:50	**00:01:23**
Cons-D	**00:00:29**	**00:00:23**	00:00:46	00:01:41
IG-D	00:00:38	00:00:28	00:00:46	00:01:43
ReliefF-D	00:00:33	00:00:26	**00:00:41**	00:02:02

In light of the above, we can conclude that our distributed proposal performs successfully, since the running time is considerably reduced and the accuracy does not drop to inadmissible values. In fact, our approach is able to match and in some cases even improve the standard algorithms applied to the non-partitioned datasets.

5 Conclusions

In this work, we have proposed a new method for scaling up feature selection: a distributed filter approach. The proposed method has been able to successfully distribute the feature selection process, shortening the execution time and maintaining the classification performance.

An experimental study was carried out on six datasets considered representative of problems from medium to large size. In terms of classification accuracy, our distributed filtering approach obtains similar results to the centralized methods, even with slight improvements for some datasets. Furthermore, the most important advantage of the proposed method is the dramatically reduction in computational time (from the order of hours to the order of minutes). As future work, we plan to distribute other FS techniques, such as wrapper or embedded methods, and to try the vertical partition instead of the horizontal one.

Acknowledgements. This research has been economically supported in part by the Secretaría de Estado de Investigación of the Spanish Government through the research project TIN 2012-37954; and by the Consellería de Industria of the Xunta de Galicia through the research projects CN2011/007 and CN2012/211; all of them partially funded by FEDER funds of the European Union. V. Bolón-Canedo acknowledges the support of Xunta de Galicia under *Plan I2C* Grant Program.

References

1. Zhao, Z., Liu, H.: Spectral Feature Selection for Data Mining. Chapman & Hall/Crc Data Mining and Knowledge Discovery. Taylor & Francis Group (2011)
2. Frank, A., Asuncion, A.: UCI Machine Learning Repository (2010), `http://archive.ics.uci.edu/ml` (accessed April 2013)
3. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature extraction: foundations and applications, vol. 207. Springer (2006)
4. Yu, L., Liu, H.: Redundancy based feature selection for microarray data. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 737–742. ACM (2004)
5. Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A.: Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. Expert Systems with Applications 38(5), 5947–5957 (2011)
6. Forman, G.: An extensive empirical study of feature selection metrics for text classification. The Journal of Machine Learning Research 3, 1289–1305 (2003)
7. Saari, P., Eerola, T., Lartillot, O.: Generalizability and simplicity as criteria in feature selection: application to mood classification in music. IEEE Transactions on Audio, Speech, and Language Processing 19(6), 1802–1812 (2011)
8. Liu, H., Motoda, H.: Feature selection for knowledge discovery and data mining. Springer (1998)
9. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics 23(19), 2507–2517 (2007)
10. Chan, P.K., Stolfo, S.J., et al.: Toward parallel and distributed learning by meta-learning. In: AAAI Workshop in Knowledge Discovery in Databases, pp. 227–240 (1993)
11. Ananthanarayana, V.S., Subramanian, D.K., Murty, M.N.: Scalable, distributed and dynamic mining of association rules. In: Prasanna, V.K., Vajapeyam, S., Valero, M. (eds.) HiPC 2000. LNCS, vol. 1970, pp. 559–566. Springer, Heidelberg (2000)

12. Tsoumakas, G., Vlahavas, I.: Distributed data mining of large classifier ensembles. In: Proceedings Companion Volume of the Second Hellenic Conference on Artificial Intelligence, pp. 249–256 (2002)
13. Das, K., Bhaduri, K., Kargupta, H.: A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. Knowledge and Information Systems 24(3), 341–367 (2010)
14. McConnell, S., Skillicorn, D.B.: Building predictors from vertically distributed data. In: Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research, pp. 150–162. IBM Press (2004)
15. Skillicorn, D.B., McConnell, S.M.: Distributed prediction from vertically partitioned data. Journal of Parallel and Distributed Computing 68(1), 16–36 (2008)
16. Rokach, L.: Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. Computational Statistics & Data Analysis 53(12), 4046–4072 (2009)
17. de Haro García, A.: Scaling data mining algorithms. Application to instance and feature selection. PhD thesis, Universidad de Granada (2011)
18. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
19. Hall, M.A.: Correlation-based feature selection for machine learning. PhD thesis, Citeseer (1999)
20. Dash, M., Liu, H.: Consistency-based search in feature selection. Artificial Intelligence 151(1-2), 155–176 (2003)
21. Zhao, Z., Liu, H.: Searching for interacting features. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, pp. 1156–1161. Morgan Kaufmann Publishers Inc. (2007)
22. Hall, M.A., Smith, L.A.: Practical feature subset selection for machine learning. Computer Science 98, 181–191 (1998)
23. Kononenko, I.: Estimating attributes: Analysis and extensions of relief. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)